CMPSC-265
Data Structures and Algorithms

Zaihan Yang
zyang13@suffolk.edu

Department of Math and Computer Science
Suffolk University

Fall 2019

Notice

- HW7 posted. Will be due on next Sunday midnight.
- Midterm_Exam1 will be held on:
  - Time: Oct 23rd (Wednesday)
  - Location: the same classroom
  - Topics:
    - The 1st week to this week's topics

Recap

- More on Quick Sort
- More on sorting algorithms

Learning Topics

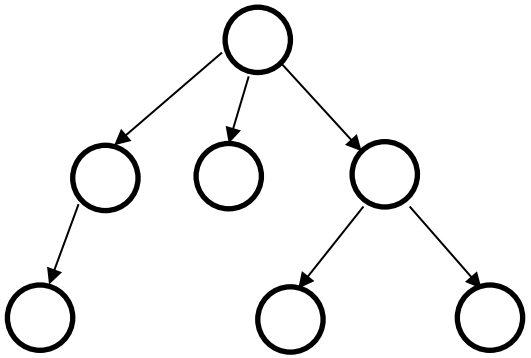
- Tree
- Binary Tree
- Basic operations of Binary Tree

Why Trees?

- Recall sorted arrays
  - Fast search, using binary search.
  - Slow insertion
- Recall linked lists
  - Efficient insertion and deletion
  - Slow search (binary search does not work)
- Tree can provide quick insertion/deletion and quick searching

What is a Tree?

- A tree is a collection of nodes connected by edges.
  - Special case of a graphs
  - It cannot contain a cycle
  - It represents a hierarchy (file system, XML…)



Tree

- A Tree data structure can be defined recursively as a collection of nodes, where each node is a data structure consisting of a data/value(item/info), together with a list of references to other nodes (the “children”).

```
public class Node {
        Item value;
        Node children1;
        Node children2;
        ……
        Node  children;
}
```

Each of such references can be null.

Definitions and Terms

- Root: the top node
- Level(depth, height): distance from root
- Parent: the node above
  - Root does not have a parent
  - Every node has one parent
- Child: the node below
  - Every node may have zero or more children
- Sibling: nodes in the same level sharing the same parent
- Leaf: node with no children
- Subtree: tree below a node

Binary Tree

- A Binary Tree is a tree data structure (a collection of nodes) where each node in addition to its value, contains references to its two children (left child and right child).

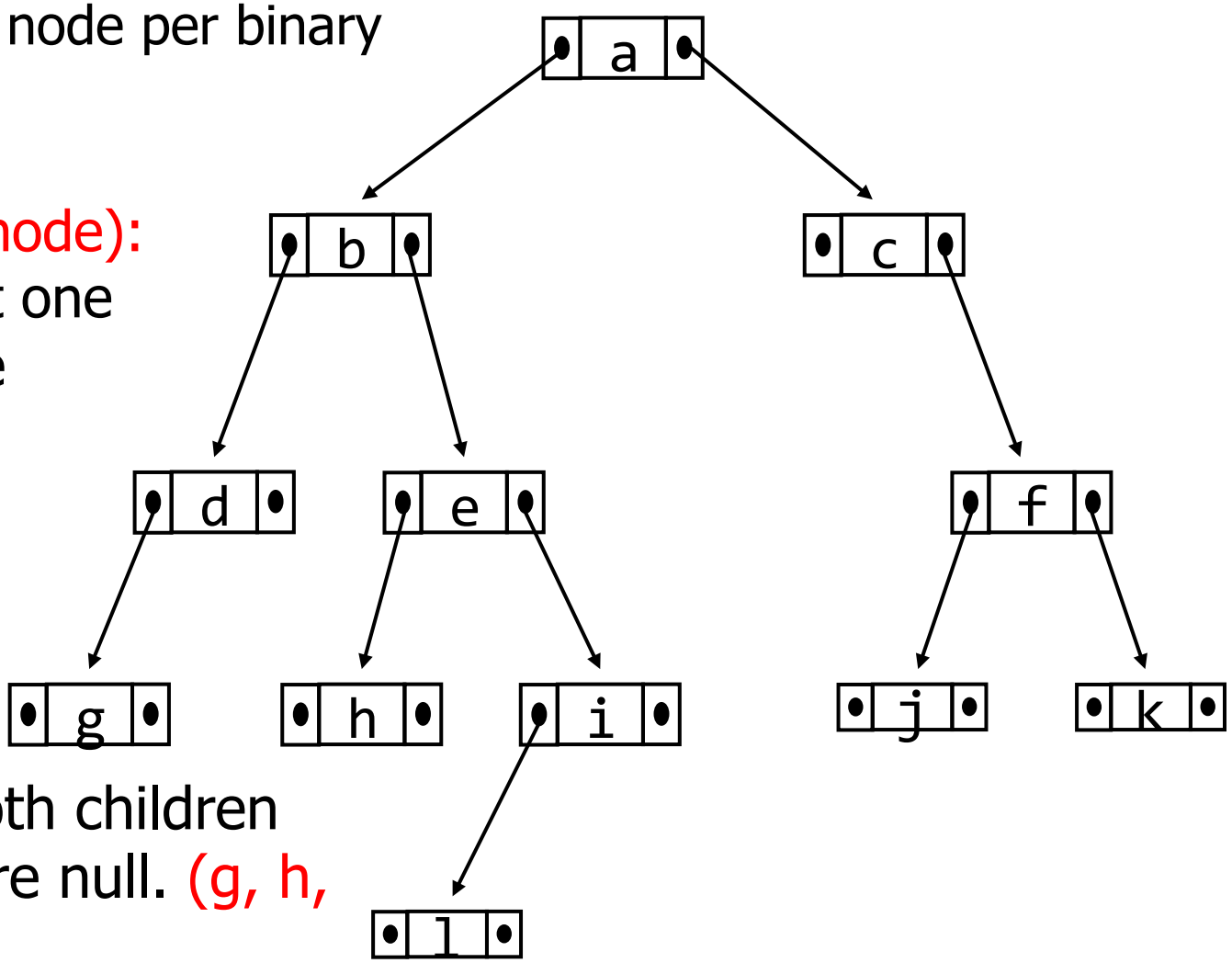
```
public class Node {
        Item value;
        Node left;
        Node right;
}
```

- A binary tree can be empty (contains no nodes)

Binary Tree: example

Root node: the topmost node. Only one root node per binary tree. (a)

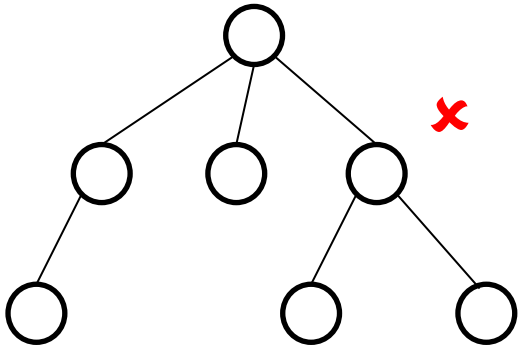
Internal (interior node): Node with at least one non-null reference (children) (b, c, d, e, f, i)

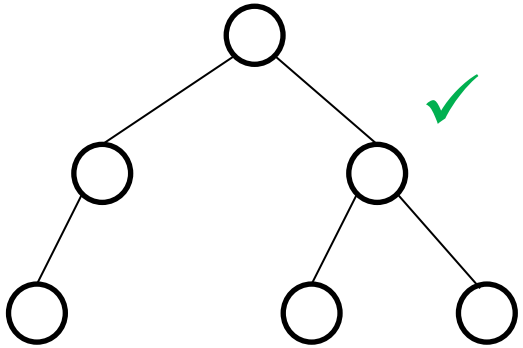


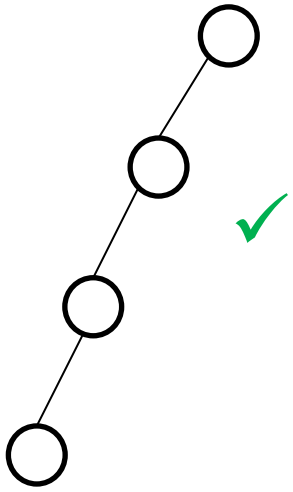
Leaf node: both children references are null. (g, h, j, k, l)

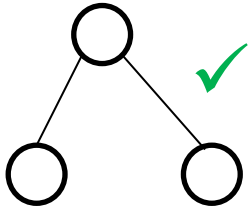
Binary Tree Examples

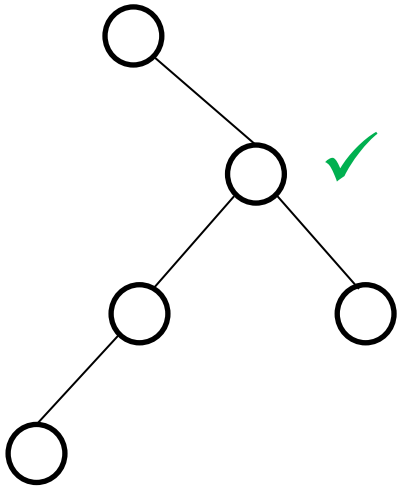
- Which one is a binary tree?





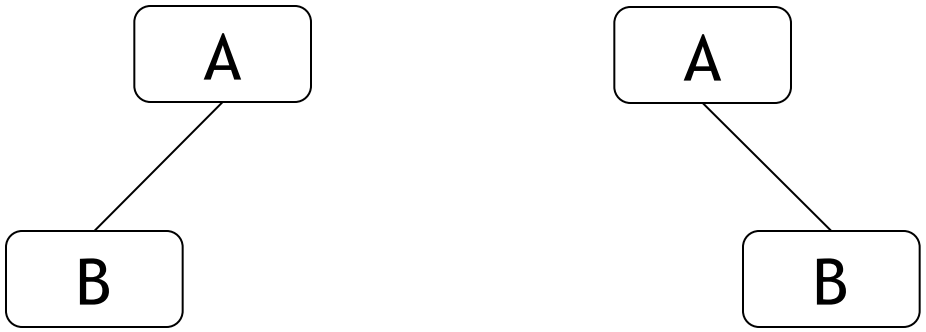






Left ≠ Right

- The following two binary trees are *different:*

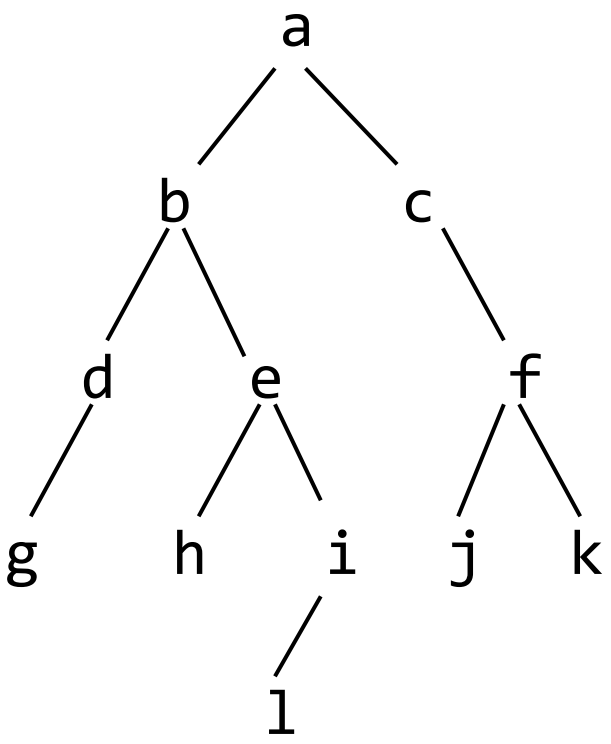


- In the first binary tree, node A has a left child but no right child; in the second, node A has a right child but no left child
- Put another way: Left and right are *not* relative terms

More terminology

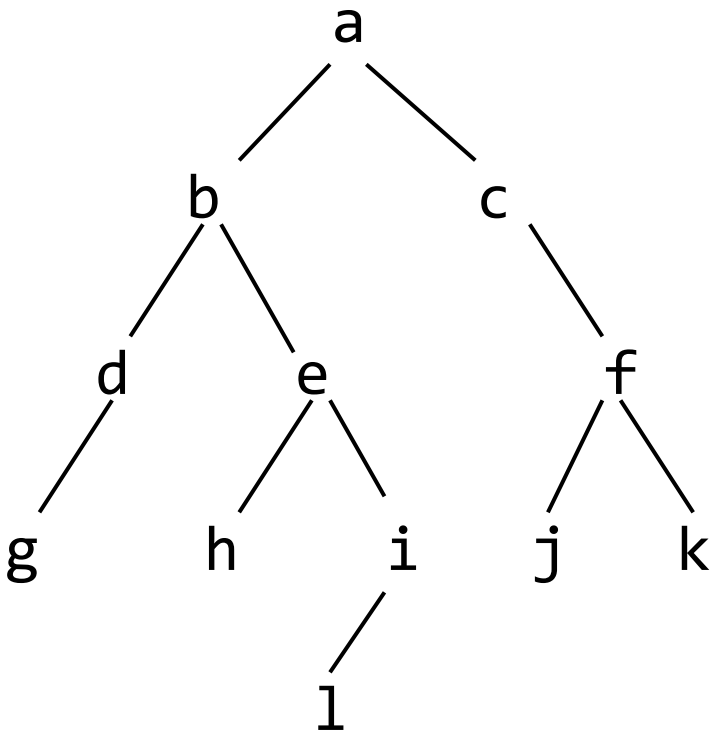
- Node A is the parent of node B if node B is a child of A.
- Root node has not parent.
- Node A is an ancestor of node B if A is a parent of B, or if some child of A is an ancestor of B
  - In less formal terms, A is an ancestor of B if B is a child of A, or a child of a child of A, or a child of a child of a child of A, etc.
- Node B is a descendant of A if A is an ancestor of B
- Nodes A and B are siblings if they have the same parent

Size and depth



- The size of a binary tree is the number of nodes in it
  - This tree has size 12
- The depth (level) of a node is its distance from the root (the number of edges from the root to the node)
  - a is at depth zero
  - e is at depth 2
- The depth of a binary tree is the depth of its deepest leaf node
  - This tree has depth 4
- The height of a node is the number of edges from the node to the deepest leaf node.
  - b is of height 3
  - Height of the tree is the height of the root

Size and depth



- At each specific level (depth) k:

  we at most of 2^k nodes;

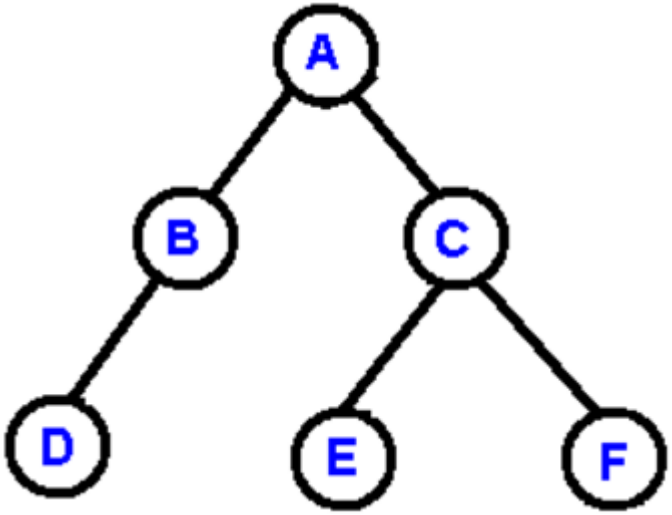
- If a Binary tree t is of N nodes, then the depth of the tree depth(t) is:

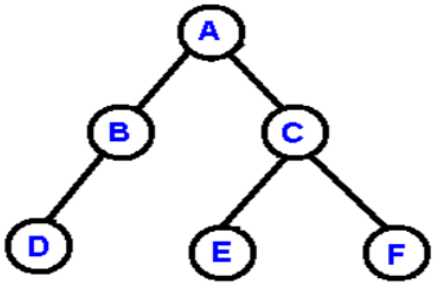
  (int)logN<=depth(t)<=N-1

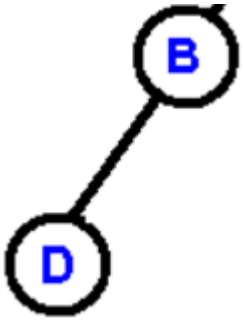
  - The left case happens when each node has two children;
  - The right case happens when each node has only one child.

Binary Tree: subtree

- A **subtree** of a binary tree *T* is a tree consisting of a node in *T* and all of its descendants in *T.* Nodes thus correspond to subtrees (each node corresponds to the subtree of itself and all its descendants)



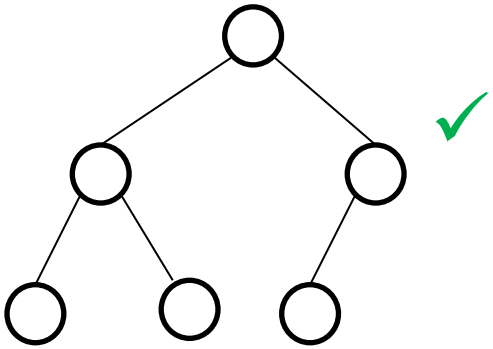


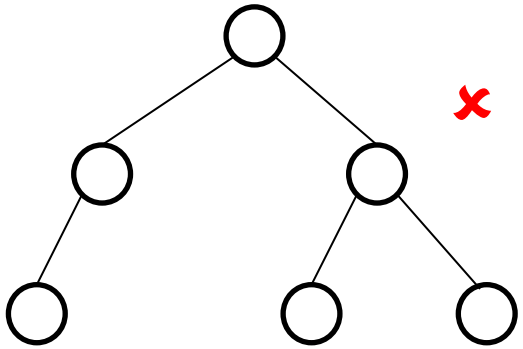


More Definitions

- Complete binary tree
  - A binary tree in which all levels, except the last level, are completely filled and all the leaves in the last level are all to the left side.

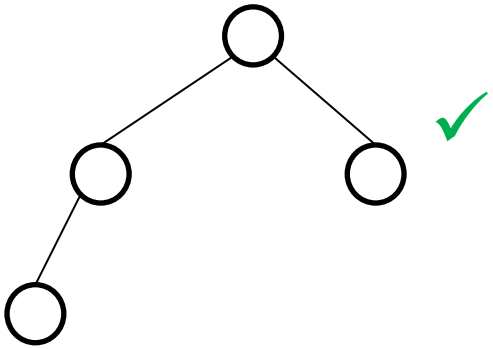
- Which one is a complete binary tree?

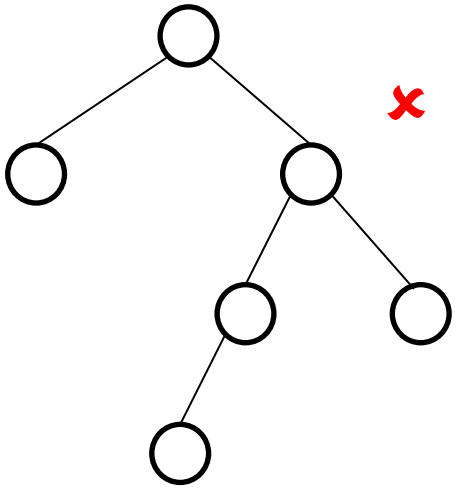




More Definitions

- Balanced binary tree
  - A binary tree where the difference between heights of the two subtrees of every node is no more than 1.
  - Which one is a balanced binary tree?



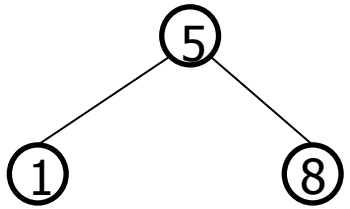


Binary Tree Node Representation

- Like the Link class in a linked list

```
class Node {
        public int key; // key value
        public Node left; // left child
        public Node right;// right child

        // constructor
        public Node(int key) {
        this.key = key;
        left = null;
        right = null;
        }
}
```

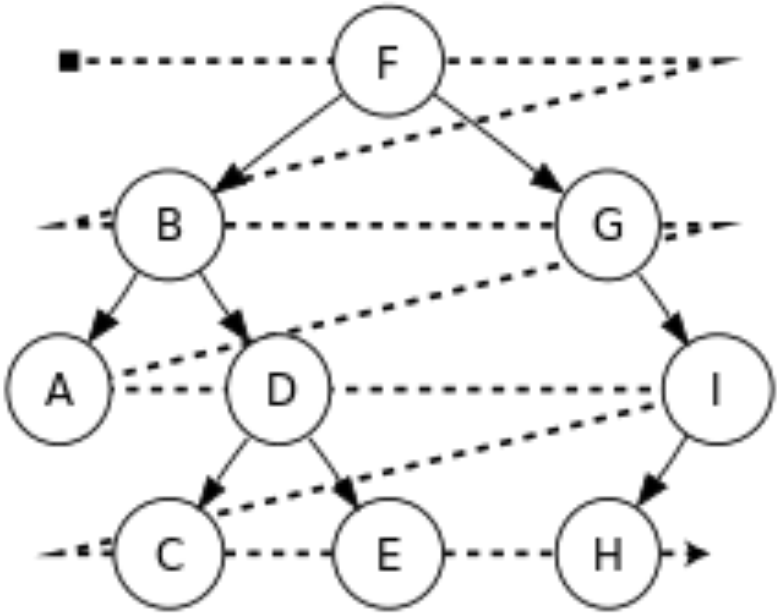


Binary Tree Operations

- To determine whether a node is a leaf node
- To get the height of a binary tree
- Traversal on a Binary Tree
  - Breadth-first traversal (level-order traversal)
  - Depth-first traversal
    - In-order
    - Pre-order
    - Post-order
- Find a path on a Binary Tree

Traversal of Binary Tree

- Breadth-first traversal (level-order traversal):
  - Visit every node on a level before going to a lower level.



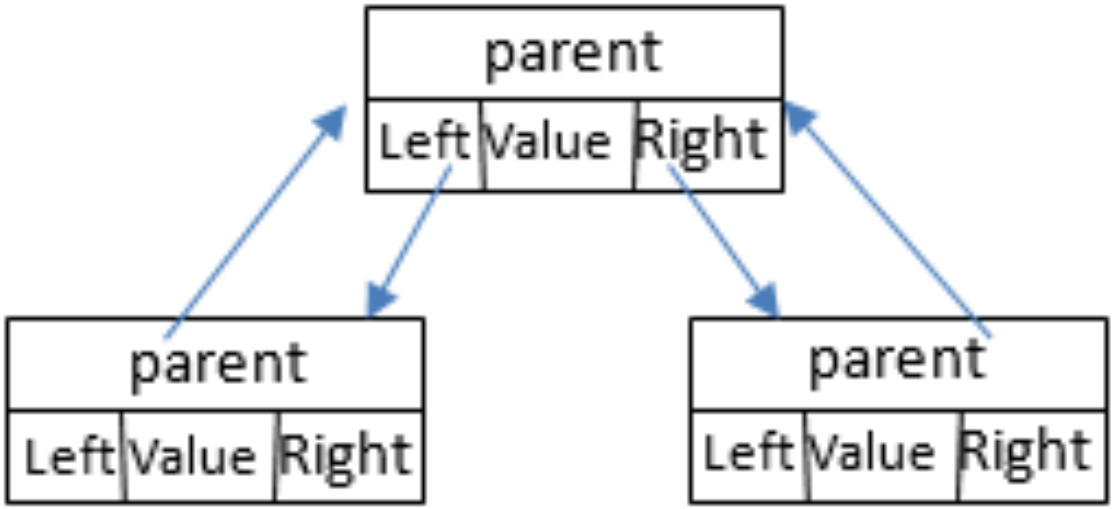
F B G A D I C E H

Traversal of the Binary Tree

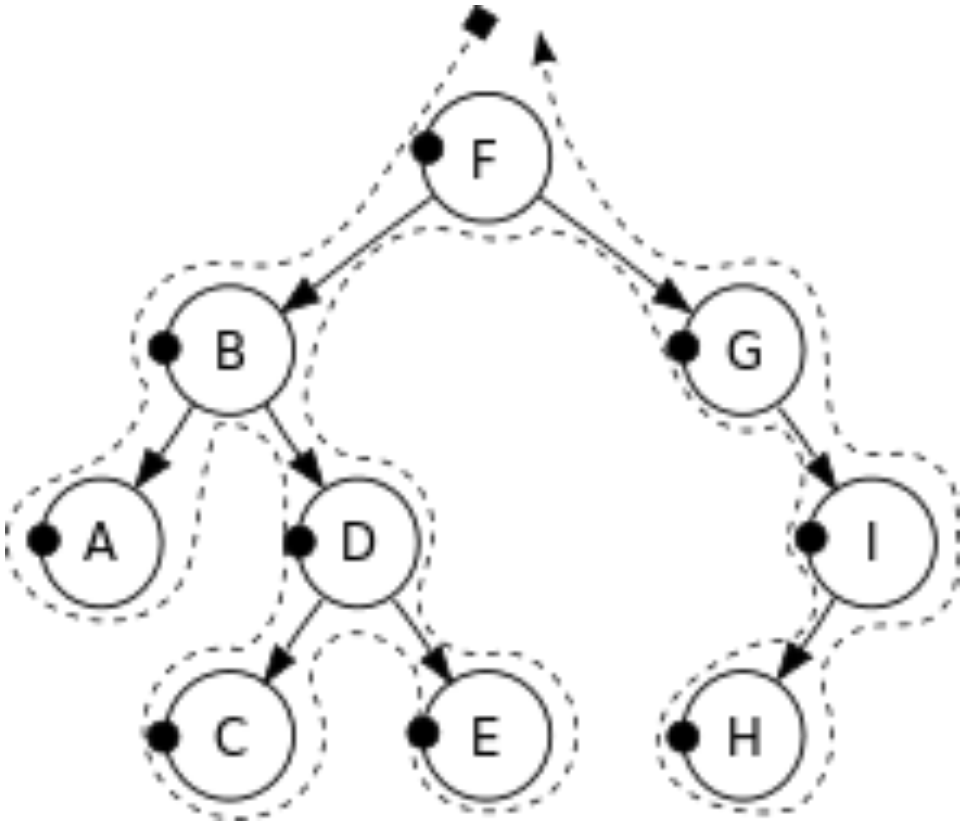
- Depth-first traversal
  - To traverse as deepened as possible on each child before going to the next sibling.
  - implemented using recursive or iterative way.
  - Three orders based upon the sequence how the root node, left child and right child is visited.
    - Pre-order: root -> recursively traverse the left subtree -> recursively traverse the right subtree.
    - In-order: recursively traverse the left subtree -> root node -> recursively traverse the right subtree
    - Post-order: recursively traverse the left subtree -> recursively traverse the right subtree -> root node.

Depth-first traversal of Binary Tree

- Suitable for representing the binary tree in a linked data structure.
- Each binary tree has a value and three references (links)



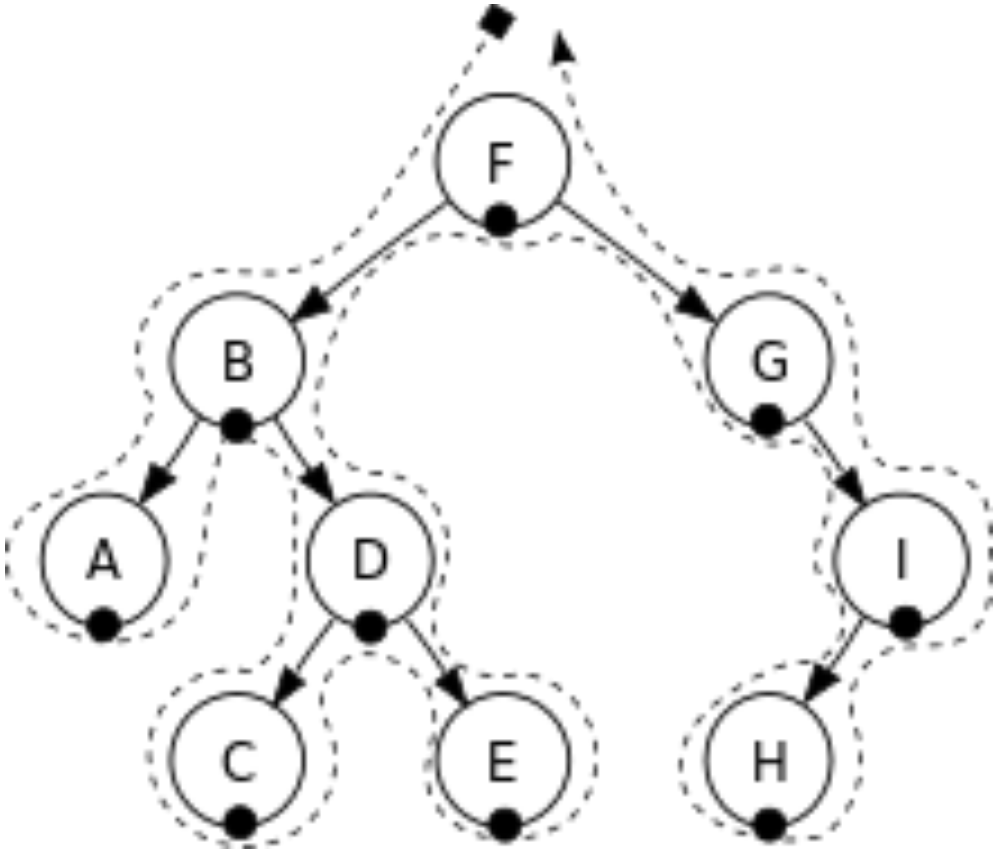
Pre-order traversal




```
preorder(node)
  if (node = null)
    return
  visit(node)
  preorder(node.left)
  preorder(node.right)
```

F B A D C E G I H

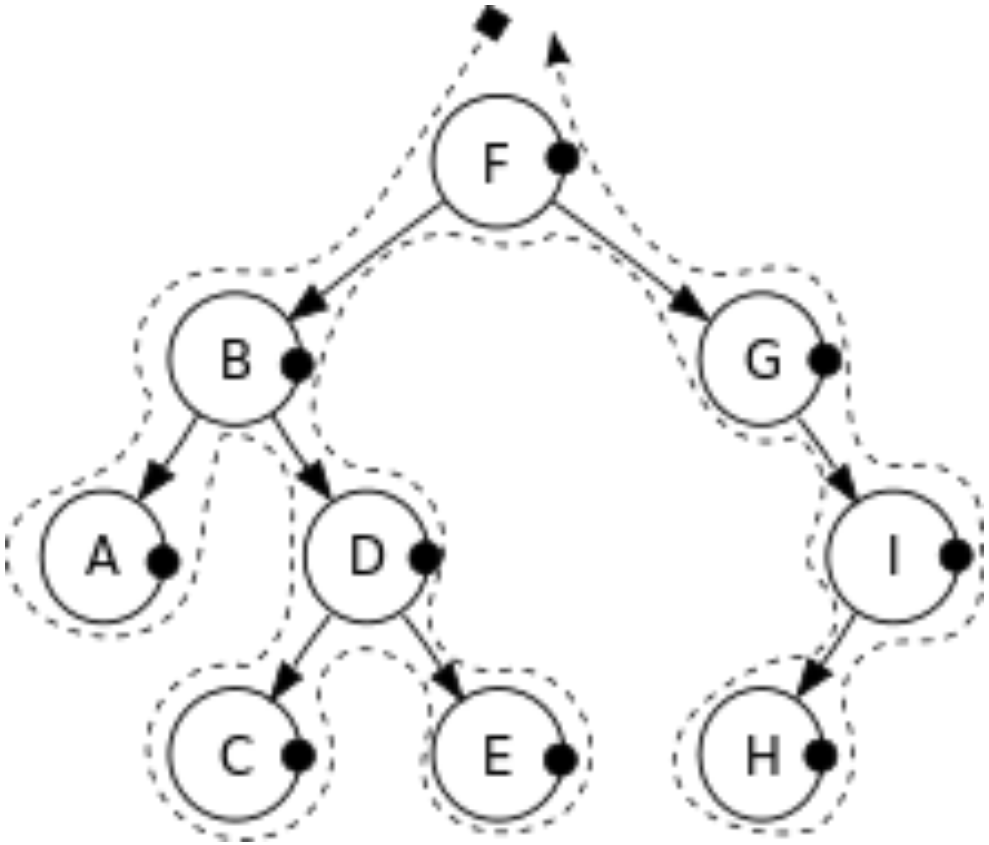
In-order traversal




```
inorder(node)
  if (node = null)
    return
  inorder(node.left)
  visit(node)
  inorder(node.right)
```

A B C D E F G H I

Post-order traversal




```
postorder(node)
  if (node = null)
    return
  postorder(node.left)
  postorder(node.right)
  visit(node)
```

A C E D B H I G F